Programación Orientada a Objetos

Mg. Alfredo Teyseyre

UNICEN

Traductor multilingüe

- Se requiere de un sistema que asista en la corrección ortográfica y en la traducción de documentos (traducción palabra a palabra).
- El sistema debe proveer los siguientes servicios:
  - traducción de un documento en varios idiomas(Ingles , Aleman, Frances, Español, Italiano) a otro idioma tomando como lenguaje intermedio al Ingles.
  - corrección ortográfica del documento previa a la traducción.
- En el proceso de traducción se cuenta sólo con traductores:
  - Ingles - otro lenguaje
  - Otro lenguaje – Ingles.

Traductor multilingüe

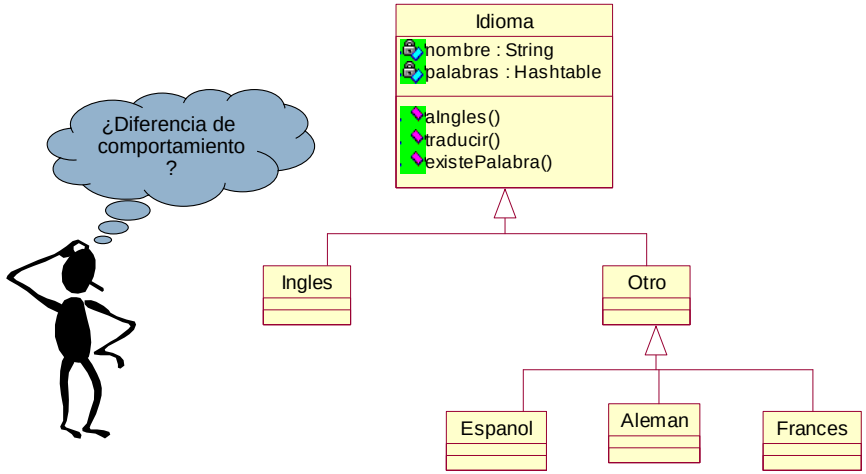
Traductor multilingüe

- **No** se deben crear **documentos intermedios**
- Antes de comenzar la traducción del documento este debe ser **corregido ortograficamente**. El proceso de corrección toma una a una las palabras del documento y si es valida se incluye en el proceso de traducción. Caso contrario esa palabra no aparece en el documento traducido y es reemplazada por “(ERROR)” en la traducción. **No** se debe crear un **documento temporal** con las correcciones.
- El documento origen define un **idioma por defecto**, pero además, **cada párrafo puede estar escrito en otro idioma distinto** al idioma por defecto del documento origen. Se debe traducir todo el documento (incluyendo sus párrafos) al idioma destino.
- Se pueden **incrustar** dentro de un documento **otros documentos**, los cuales también deben ser traducidos.
- El documento traducido debe **respetar la estructuración del documento origen** tanto para los parrafos como para los documentos incrustados.

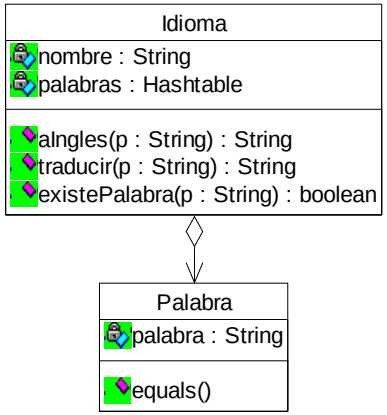
¿Qué objetos aparecen?

- De una primera lectura surgen los siguientes objetos:
  - Documento
  - Párrafos
  - Palabras
  - Idiomas
  - Traductor
  - Corrector

Idiomas



Idiomas



Idiomas

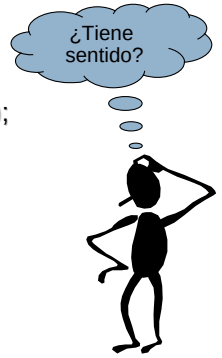
```
public class Idioma {

   private Hashtable palabras;
   String nombre;

   public Idioma(String nombre) {
            this.nombre = nombre;
            palabras = new Hashtable();
   }
   public void addPalabra(Palabra origen, Palabra enIngles) {
            palabras.put(origen, enIngles)
   }
   public boolean existePalabra(Palabra p) {
      return palabras.containsKey(p);
   }
…
```

Idiomas

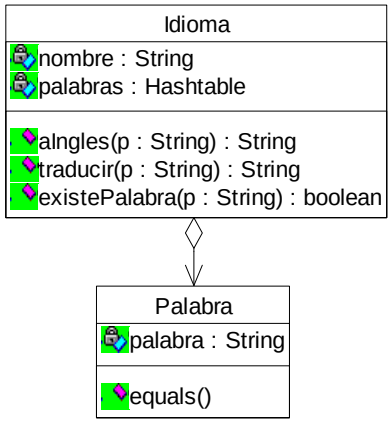
```
…

   public Palabra aIngles(Palabra p) {
            if (existePalabra(p))
                     return (Palabra) palabras.get(p);
   }

   public Palabra traducir(Palabra p) {
            Palabra traducida;
            for (Enumeration e = palabras.keys() ;
                     e.hasMoreElements() ;){
                traducida = (Palabra) e.nextElement();
                if (palabras.get(traducida).equals(p))
                     return traducida;
            }
            return null;
   }

}
```


Palabras

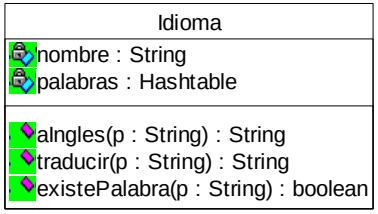
```
public class Palabra {

   private String palabra;

   public Palabra(String p) {
            palabra = p;
  }

  public boolean equals(Palabra p) {
            return palabra.equals(p.getPalabra());
  }

  public String getPalabra(){
            return palabra;
  }
}
```



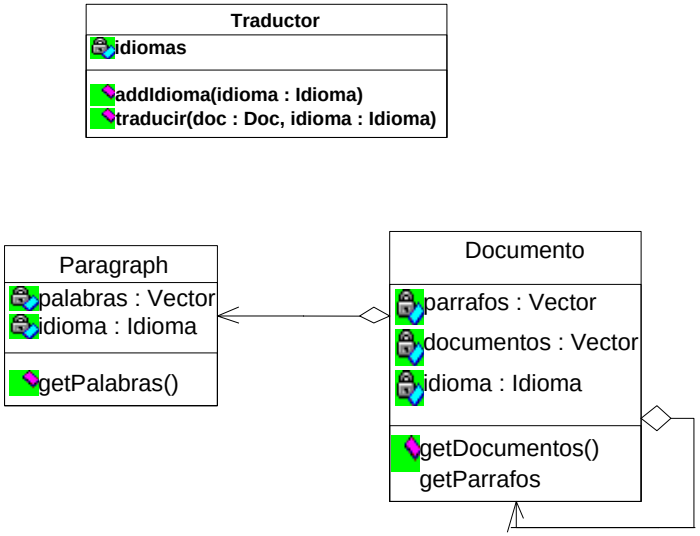
Idioma



Idiomas



Traductor



Documento

```
public class Documento {
    private Vector parrafos;
    private Vector documentos;
    private Idioma idioma;

    public Documento(Idioma i) {
            parrafos = new Vector();    documentos = new Vector();}

    public addParrafo(Parrafo p) {
            parrafos.add(p);    }

    public addDocumento(Documento p) {
            documentos.add(p);    }

    public Vector getParrafos {
            return parrafos;    }

__public Vector getDocumentos {
            return documentos;    }
```

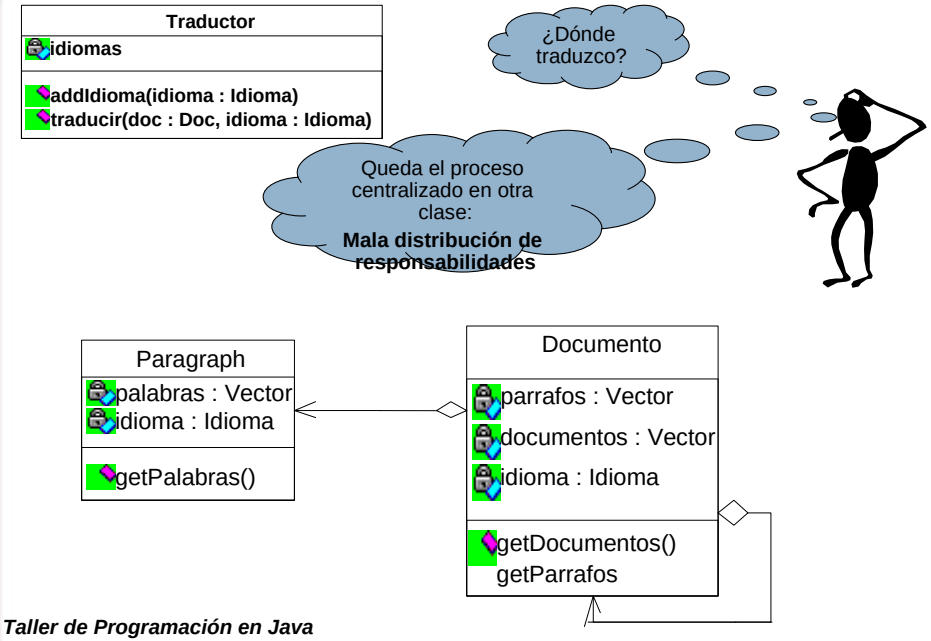

Párrafo

```
public class Parrafo {
    private Idioma idioma;
    private Vector palabras;

    public Parrafo(Idioma i) {
            palabras = new Vector();
}
    public addPalabra(Palabra p) {
            palabras.add(p);
    }

    public Vector palabras(){
            return palabras;
    }
…
```

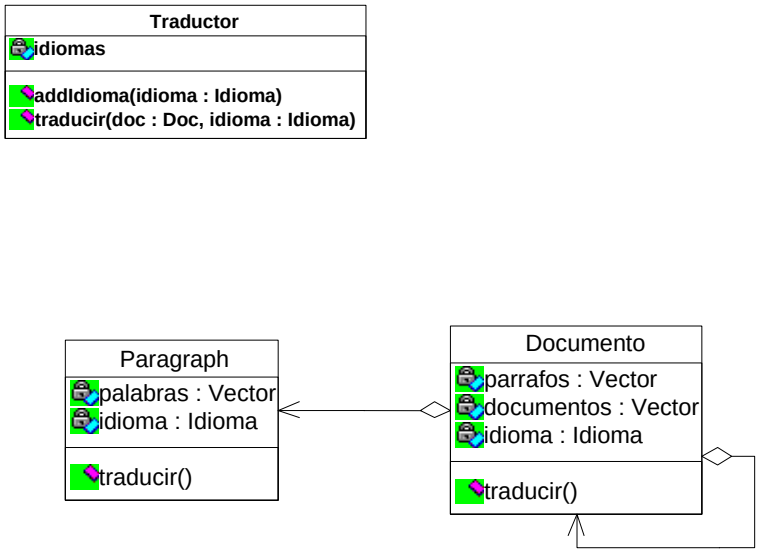

Traductor

```
public Documento traducir(Documento doc,Idioma destino) {
  Documento d = new Documento(destino);  Parrafo p;
  for (Enumeration e=doc.getParrafos().elements();e.hasMoreElements() ;
            p = (Parrafo) e.nextElement()) {
            Parrafo np = new Parrafo(destino)
            String pal;            String palIngles
            for (Enumeration e = p.palabras().elements() ;e.hasMoreElements() ;
                  pal = (String) e.nextElement()) {
                  if (p.idioma().existe(pal)) {
                              palIngles = p.idioma().aIngles(pal)
                              np.addPalabra(destino.traducir(palIngles));    }
                        else  np.addPalabra("ERROR");
                  }
            }
            d.addParrafo(np);
    }
 return d;  // Falta traducir documentos incrustados
                  (llamar a traducir con c/ documento)
```

Traductor



Traductor



Documento

```
public class Documento {
    private Vector parrafos;
    private Vector documentos;
    private Idioma idioma;

    public Documento(Idioma i) {
            parrafos = new Vector();    documentos = new Vector();}

    public add (Parrafo p) {
            parrafos.add(p);    }

    public add (Documento p) {
            documentos.add(p);    }

    public Vector getParrafos {
            return parrafos;    }

__public Vector getDocumentos {
            return documentos;    }
```


Documento

```
public Documento traducir(Idioma destino) {
            Documento nd = new Documento(destino)
            Texto t;
            for (Enumeration e=parrafos.elements();e.hasMoreElements())  {
                        Parrrafo p= (Parrafo) e.nextElement())
                nd.add (p.traducir(i));}
            for (Enumeration e=documentos.elements();e.hasMoreElements() ){
                        Documento d= (Documento) e.nextElement())
                nd.add (d.traducir(i));
            }
            return nd;
}
```

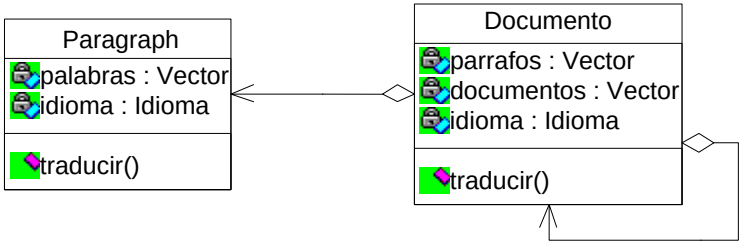
Párrafo

```
public class Parrafo {
    private Idioma idioma;
    private Vector palabras;

    public Parrafo(Idioma i) {
            palabras = new Vector();
}
    public addPalabra(Palabra p) {
            palabras.add(p);
    }

    public Vector palabras(){
            return palabras;
    }
…
```

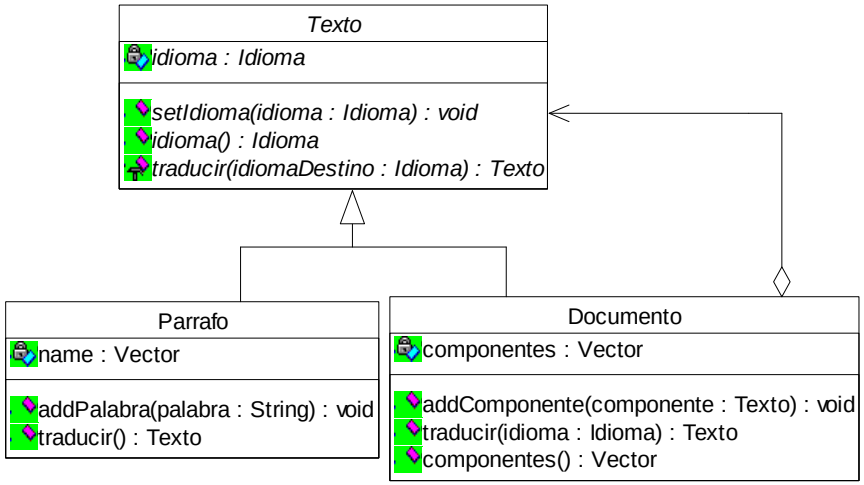

Párrafo

```
public Parrafo traducir (Idioma destino) {
        Parrafo p = new Parrafo(destino)
        String pal;          String palIngles
        for (Enumeration e = palabras().elements() ;e.hasMoreElements() ) {
            pal = (String) e.nextElement())
            if (idioma().existe(pal)) {
                palIngles = idioma().aIngles(pal)
                p.addPalabra(destino.traducir(palIngles));
            }
            else p.addPalabra("ERROR");
        }
        return d;
    }
}
```


Texto



Texto



Texto

```
public abstract class Texto {

    private Idioma idioma;

    public Texto(Idioma i) {
            idiomaDefecto = i;
}
    public Idioma idioma() {
            return idioma;
    }

    public abstract Texto traducir(Idioma destino);

}
```


Documento

```
public class Documento extends Texto{

    private Vector componentes;

    public Documento(Idioma i) {
            super(i);
            componentes = new Vector();
}
    public addComponente(Texto t) {
            componentes.add(t);
    }

    public Vector componentes(){
            return componentes;
    }
…
```


Documento

```
…

    public Texto traducir(Idioma destino) {
            Documento d = new Documento(destino)
            Texto t;
            for (Enumeration e = componentes().elements() ;
                    e.hasMoreElements() ;
                    t = (Texto) e.nextElement())
                d.addComponent(t.traducir(i));

            return d;
    }
}
```


Párrafo

```
public class Parrafo extends Texto{

    private Vector palabras;

    public Parrafo(Idioma i) {
            super(i);
            palabras = new Vector();
}
    public addPalabra(Palabra p) {
            palabras.add(p);
    }

    public Vector palabras(){
            return palabras;
    }
…
```

Párrafo

```
…

    public Texto traducir (Idioma destino) {
            Parrafo p = new Parrafo(destino)
            String pal;
            String palIngles
            for (Enumeration e = palabras().elements() ;
                        e.hasMoreElements() ;
                        pal = (String) e.nextElement()) {
                if (idioma().existe(pal)) {
                    palIngles = idioma().aIngles(pal)
                    p.addPalabra(destino.traducir(palIngles));
                }
                else
                    p.addPalabra("ERROR");
            }
            return d;
    }
}
```


Juntando todo…